

1 METHOD AND SYSTEM FOR CONTROLLING AND FILTERING FILES USING A
2 VIRUS-FREE CERTIFICATE

3 *TECHNICAL FIELD*

4 The present invention relates to computer viruses and more
5 particularly to a method and system for controlling and
6 filtering files using an virus-free certificate.

7 *BACKGROUND ART*

8 Among all computing and networking security issues, the most
9 important cause of concern does not come from intrusions, but
10 from the widespread proliferation of viruses. Viral infections
11 represent the great majority of all security incidents.

12 **Virus Protection**

13 Virus protection for large organizations has become more and
14 more complex and difficult because of :

- 15 • the combined use of heterogeneous systems and practices,
16 • the widespread use of distributed or client/server systems,
17 and
18 • the free exchange of data files via network sharing,
19 e-mail, Internet ...

20 Until recently, viral infections threatened only data residing
21 on storage media, such as hard drives and floppy disks.

22 However, with the emergence of macro viruses, the threat has
23 spread to applications. Most organizations are not aware of

1 this level of penetration and are not organized to manage and
2 prevent virus attacks. An effective virus protection software
3 must prevent infections rather than simply treating them after
4 they have already occurred. Anti-virus solutions need a
5 uniform plan, with a centralized control, automated virus
6 signature updates, and support for multiple platforms,
7 protocols, and file types.

8 **Computer Viruses**

9 A computer virus is any program created to reproduce itself. A
10 virus reproduces itself by attaching itself to programs,
11 files, or even to boot sectors of disks. A virus is activated
12 when the infected file or disk is opened or accessed. Once a
13 virus resides in a memory, it can attach itself to the next
14 file or disk accessed, and so on. A virus may be designed to
15 do harm. A virus may also have unintended consequences by
16 overwriting important computer information and by causing
17 costly inconveniences to users and network managers. There are
18 four general types of computer virus:

- 19 • **File Viruses** (including macro viruses), which are attached
20 to files;
- 21 • **Boot sector Viruses** in which the boot sectors of floppy or
22 hard disks are infected;
- 23 • **Master Boot Record (MBR) Viruses** which infect the disk
24 master boot record; and
- 25 • **Multi-partite Viruses** that are a combination of a file
26 virus and a boot sector virus.

27 **Virus Disguises**

1 Viruses need to avoid detection in order to succeed in
2 corrupting target computers. Simple viruses, with easily
3 detectable signatures are giving way to more sophisticated
4 virus types:

- 5 • **Polymorphic Viruses** : they change their signature, or
6 profile, each time they are activated so that a fixed
7 signature filter will miss them.
- 8 • **Stealth Viruses** : they attempt to hide their presence by
9 intercepting interrupt services and by feeding back false
10 information to anti-virus products and end users.
- 11 • **Encrypted Viruses** : they are delivered within an
12 encrypted file and are undetectable by a simple
13 anti-virus.

14 .

15 **Sources of Infection**

16 Every improvement in network and communication technologies
17 opens new avenues through which viruses can infect your
18 system. Most of former viruses were boot sector viruses, in
19 which the boot sectors of floppy or hard disks were infected.

20 **Macro Viruses**

21 As stated earlier, the creation of macro viruses has changed
22 this environment dramatically. A macro virus is a set of
23 instructions comprising powerful macro routines initially
24 designed for word processing and spreadsheet applications.
25 These macro languages enable a myriad of useful functions
26 which can be imbedded into a document and which can be
27 executed when the document is opened for view or use.

28 **Internet**

1 With the exploding development of the Internet, viruses have
 2 catastrophic possibilities. The Internet introduces two
 3 different virus threats.

- 4 • The first threat is caused by the download of files
 5 comprising viruses when these files are browsed or
 6 transferred using for instance FTP (File Transfer Protocol)
 7 routines. Public shareware (shared software) and executable
 8 routines of all types, including formatted presentations,
 9 are a growing source of virus infection. Furthermore, new
 10 Internet virus threats are beginning to appear in the form
 11 of malicious JAVA and Active-X applets.
- 12 • The second threat comes from electronic mail (e-mail). Most
 13 Internet e-mail systems provide a very rich capability to
 14 attach formatted documents to mail sent over the network.
 15 These e-mail messages can be broadcast to individuals or
 16 groups of individuals with the simple stroke of a key!
 17 Infected documents or files can flood a corporate network
 18 through gateways and mail servers. As networking,
 19 telecommunications, remote access, message systems
 20 supporting attachments of all kinds become more and more
 21 common, viruses will exploit these new electronic pathways
 22 to attack systems that were heretofore unreachable.

23 **Groupware Complications**

24 A third trend in networking also exacerbates the virus threat:
 25 the trend towards the deployment of Groupware applications
 26 such as Lotus Notes, Microsoft Exchange, Novell Groupwise,
 27 etc.

28 Since the active and repeated sharing of documents over the
 29 network is at the core of these applications, they represent a

1 fertile ground for the deployment of macro viruses. A
2 Groupware application not only acts as a repository for shared
3 documents, but, due to its collaborative function, it
4 simultaneously broadcasts files to associated work groups. The
5 broadcast of files significantly multiplies the possibility of
6 accidentally deploying mail infected by attached macro viruses
7 and makes Groupware protection a high priority.

8 **Symptoms of Virus Infection**

9 Most viruses attempt to remain undetected as long as possible
10 to extend their destructive influence. Therefore, most viruses
11 do not produce any recognizable profile or signature that
12 would allow to trap them by scanning the software. However,
13 viruses perform actions that do not look like normal computer
14 operations or user operations. These abnormal actions can be
15 detected by intelligent anti-virus software. Fortunately, many
16 viruses have telltale symptoms and may inadvertently give off
17 signals that can alert users and virus protection software to
18 their presence.

19 Some of these symptoms include:

- 20 • Increase in byte length of files,
- 21 • Alterations of a file's time stamp,
- 22 • Delayed program loading or activation,
- 23 • Reduced performance,
- 24 • Lower system resources, available memory, disk space,
- 25 • Bad sectors on floppies and hard drives,
- 26 • Strange or non-standard error messages,
- 27 • Non-standard screen activity, display fluctuations,
- 28 • Program inoperability (failing to execute),
- 29 • Incomplete or failed system boots, and
- 30 • Uninitiated drive writes.

1 ANTI-VIRUS SOFTWARE OVERVIEW

2 Detecting a Virus

3 Viruses are becoming increasingly sophisticated and, as such,
4 can defeat simpler, single dimension software packages. To be
5 effective, the anti-virus software must include
6 special-purpose, distributed applications. Applications can
7 detect viruses using five distinct methods:

8
9 • **Signature Scanning:** This method compares the content of
10 files against a database of virus signatures. This method
11 requires frequent updates of the database to ensure the
12 identification of new and changing signatures.

13 • **Integrity Checking:** This method compares the profile of
14 current files and disk areas against an archived snap shop
15 of these same items. The detected differences may indicate
16 the presence of a virus. Check summing is the most common
17 type of integrity checking. Unfortunately, integrity
18 checking is generally not effective against modern stealth
19 viruses, so further detecting means are needed.

20 • **Heuristic Analysis:** An artificial intelligence monitors
21 virus-like behavior, such as trapping certain interrupt
22 services or attempting unlikely actions such as
23 reformatting the hard disk.

24 • **Polymorphic Analysis:** Polymorphic viruses are difficult
25 to detect because they constantly change their look,
26 particularly when they are encrypted or when they use
27 stealth techniques to hide their presence. A polymorphic
28 analyzer will move any suspect file to a separate,

1 protected, location in the computer and will execute it
2 there to see if it exhibits any virus-like behavior.

3 • **Macro Virus Analysis:** A specifically designed anti-virus
4 software detects macros in files and tests them before
5 execution.

6 **Archived and Compressed Files**

7 In addition to the support of these five types of virus
8 analysis, an effective anti-virus system must also be able to
9 scan archived and compressed files. Zip (or Pkzip) and
10 Microsoft Compression are the most common tools for archiving
11 and compressing a file. A virus can hide inside a compressed
12 archive, and can remain dormant or unnoticed until the
13 infected file is extracted and released into a system. The
14 minimum for an efficient anti-virus system is to be able to
15 scan most current types of archives to identify viruses stored
16 within the files they contain.

17 **Frequency of Database Signature Update**

18 Finally, the ability of a virus software to prevent virus
19 attacks is determined by its ability to maintain an updated
20 virus signature database. Any anti-virus software must have an
21 associated, easily accessible Web site, or some other online
22 source of information, where regular virus database updates
23 can be retrieved. Products that automate this update process
24 by using an Internet connection to regularly collect new
25 information have a clear advantage in this regard.

26 **Real Time and Scheduled Virus Scanning**

27 Most anti-virus software can perform a scan of a computer in
28 order to detect and possibly treat the viruses found at that

1 time. This process is called scanning. Scanning a computer for
2 viruses can occur :

- 3 • at regular intervals under the control of a scheduler, or
- 4 • as an on-demand operation manually executed, or
- 5 • as an event-activated operation (usually in response to
- 6 some recognizably "illegal" behavior by a potential virus).

7 In addition, viruses can be detected in real time, when they
8 are received. This capability is important because if viruses
9 can be detected when they attempt to enter within a system
10 (computer, data repository, server ...), then it is possible
11 to prevent them from corrupting other files. Oftentimes, a
12 scheduled scan may occur after

13 a virus has already entered within a computer and has
14 corrupted other files. Obviously, the earlier a virus can be
15 detected, the better.

16
17 To be truly useful, an anti-virus software must have the
18 ability to perform all types of scans.

19 **Certificate**

20 A Certificate is a structure that contains a public value
21 (i.e. a public key) associated with an identity. For instance,
22 within a X.509 Certificate, the public key is bound to a
23 "user's name". A third party (a Certificate Authority) attests
24 that the public key belongs to the user. A X.509 Certificate
25 is a very formal structure and comprises different elements:

- 26 • **Subject:** This is the "user's name" (the Subject can be
- 27 any identity value).

- 1 • **Issuer:** This is the name of the third party that has
2 issued/generated the certificate. This third party is the
3 Certificate Authority (CA).
- 4 • **Public Key Value:** This is the public key of a
5 public/private key pair. An associated field defines the
6 public key algorithm that must be used, for instance a
7 RSA , Diffie-Hellman or DSA public key.
- 8 • **Validity:** Two fields are used to define the period of
9 validity (valid from date 1 and valid to date 2).
- 10
- 11 • **Serial Number:** This field provides a unique Certificate
12 serial number for the issuer.
- 13 • **Signature:** The signature is an encrypted digest generated
14 by the Certificate Authority (CA) for authenticating the
15 whole certificate. The digest results from the hashing of
16 the Certificate. The digest is encrypted using the CA
17 private key. The encrypted digest which is the signature,
18 "certifies" that the Subject is the "owner" of the public
19 and private keys.

20 **Certificate Verification**

21 The Certificate needs to be verified to ensure that it is
22 valid. This is a quite complex process. The verification by an
23 end user of a Certificate comprises the checking of the
24 following elements:

- 25 • Valid (or any) Subject and Issuer names are defined in
26 the Certificate.

- 1 • The Certificate is not expired (checking of the Validity
2 period field).
- 3 • The Certificate has not been revoked (this may be
4 determined by obtaining a current Certificate Revocation
5 List from the CA).
- 6 • The signature on the Certificate is valid (the signature
7 is not verified by using the Certificate's public key but
8 by using the CA public key).

9 The method for validating the signature is quite simple, and
10 comprises the steps of:

- 11 • extracting the issuer's name (CA name) from the
12 Certificate;
- 13 • locating the issuer's Certificate (CA Certificate) or
14 the issuer's public key (CA public key).
- 15 • checking that the end user's Certificate signature was
16 generated by the issuer (CA) using the issuer's public
17 key (CA public key).

18 Certificates are generated by a Certificate Authority (CA).

19 Two main methods can be used:

- 20 • **Centralized Generation:** The private/public key pair is
21 generated by the end user (defined in the subject field of
22 the Certificate). The public key is directly provided by
23 the end user to the CA software to create a Certificate.
24 The Certificate can be provided to another end user via any
25 suitable channel. The channel does not have to be secure
26 because a Certificate is a self protecting structure (given
27 the CA's signature).

1 • **Distributed Generation:** The private/public key pair is
2 generated by the end user. The end user requests the CA to
3 build a Certificate including the end user public key. The
4 public key is then sent to the CA for certification. If the
5 request is valid then the CA returns a Certificate
6 associating the user identity with the user public key to
7 the end user.

8 Of course these two methods can be combined in any system,
9 because trusted CA keys are generated by the Certificate
10 Authority (CA).

11 **OBJECTS OF THE INVENTION**

12 Current anti-virus method are becoming more and more complex
13 due to:

- 14 • the number of viruses,
- 15 • the difficulty to find them, and
- 16 • the fact that their signature can change with time or
17 environment.

18 Virus are coming from everywhere and especially from the
19 Internet network. The time required to check a disk within a
20 computer system, becomes more and more important. Furthermore,
21 the checking of a disk involves the use of resources which may
22 prevent the normal use of the computer system.

23 It is an object of the present invention to improve current
24 anti-virus checking methods and to provide a new method using
25 file Certificates similar to X.509 Certificates used to
26 authenticate an identity. A specific process associates a

1 Certificate, called virus-free Certificate (VC), with a file
2 in order to speed up and improve the virus detection.

3 It is another object of the present invention is to reduce the
4 consumption of resources (for instance, the CPU - Central
5 Processing Unit) and to reduce the time necessary to detect
6 viruses within files. This reduction is especially important
7 on systems handling a huge amount of traffic (for instance IP
8 Routers or Firewalls). The performance of such systems is
9 highly impacted by usual anti-virus checkers because usual
10 checkers require to process each file. The detection of
11 viruses on said systems is a very complex process and must be
12 done as fast as possible.

13 It is another object of the invention, when a system requires
14 a virus-free Certificate for a particular file, to retrieve
15 this virus-Free Certificate as fast as possible, for obvious
16 performance reasons. Copies of existing virus-free
17 Certificates are stored instead of being rebuilt each time
18 they are required. Retrieving an existing Virus-free
19 Certificate is more efficient (saves time and provides better
20 performances) than rebuilding a new virus-free Certificate.
21 Within a network, multiple authorities can be used to build a
22 virus-free Certificate for a particular file. They can share,
23 for instance, the load of building virus-free Certificates and
24 can use different anti-virus checkers. An authority among
25 these multiple authorities must be identified when a
26 virus-free Certificate has to be retrieved.

27 *SUMMARY OF THE INVENTION*

28

1 The present invention relates to computer viruses and
2 discloses a method and system for controlling and filtering
3 files using a virus-free Certificate. The virus-free
4 Certificate associated with each file comprises in a file
5 signature. The signature is generated by a trusted server (a
6 Virus-free Certificate Authority - VCA) which avoids the
7 system which receives the file to check this file for all
8 existing virus. The Virus-free Certificate Authority validates
9 the file against all known viruses, using one or several
10 anti-virus checkers. In case of new viruses, only the
11 virus-free Certificates is changed and the only process
12 performed by the system receiving the file (typically a
13 network device) is to verify the file against the signature
14 included in the virus-free Certificate, and to filter the file
15 according to predetermined rules. The present invention
16 drastically simplify the computing resources used for
17 detecting viruses on network devices such as IP Routers and
18 Firewalls. Files on Web Servers can be downloaded with their
19 anti-virus Certificates suppressing the risk of virus. The
20 full anti-virus is processed once on the Virus-free
21 Certificate Authority instead of being processed on each
22 system. Since the processing resources required on the system
23 are limited (because the anti-virus checker is processed on
24 the Virus-free Certificate Authority and not on the network
25 device), the performance impact on the system is also limited.

26 An example embodiment of the present method comprises the
27 steps of:

28 • receiving a file;

29 If a virus-free certificate is required for the file:

```
1  * determining whether the a virus-free certificate is already
2  associated with the file;
```

```
3 If a virus-free certificate is already associated with the
4 file:
```

5 • authenticating the associated virus-free certificate, said
6 virus-free certificate comprising a certificate signature;

7 If the virus-free certificate is authenticated :

- 8 • determining whether the file is virus-free or not:

9 If the file is virus-free:

```
10  • forwarding the file with the associated virus-free
11  certificate;
```

```
12 If the virus-free certificate is not authenticated or if no
13 virus-free certificate is associated with the file:
```

```
14  ◦ determining whether the file is virus-free or not:
```

```
15 If the file is virus-free:
```

```
16  • associating with the file a new virus-free certificate;
```

```
17  ◦ forwarding the file with the new virus-free certificate.
```

19 BRIEF DESCRIPTION OF THE DRAWINGS

20 The novel and inventive features believed characteristics of
21 the invention are set forth in the appended claims. The
22 invention itself, however, as well as a preferred mode of use,
23 further objects and advantages thereof, will best be
24 understood by reference to the following detailed description
25 of an illustrative detailed embodiment when read in
26 conjunction with the accompanying drawings, wherein :

27 • Figure 1 describes the different entities involved in the
28 anti-virus system according to the present invention;

SECRET

23 PREFERRED EMBODIMENT OF THE INVENTION

25 Figure 1 describes the different entities involved in the
26 virus system disclosed in the present invention. In most of
27 the cases, the file that the Client Workstation (101) requires
28 is stored in a File Server (for instance a WEB Server) (103).
29 A Certificate is associated with this file on the File Server

(103). The Certificate indicates that the file has been processed by a list of anti-virus programs (also referred to as anti-virus checkers), and is virus free. It is an object of the Virus-free Certificate Authority (VCA) (104) to build such Certificates (called Virus-free Certificates - VC). The Client Workstation, the File Server, the Virus-free Certificate Authority are attached to a LAN / WAN network (102) (Local Area Network / Wide Area Network). which can include the Internet network.

7

The Client workstation (101) downloads the file and the associated anti-virus Certificate from the File Server (103) through a network device within the WAN/LAN network. Said network device is for instance:

- an IP (Internet Protocol) Router which routes files between File Servers and Client Workstations,
- a Firewall protecting the secure side of the LAN/WAN network (typically the Client Workstations attached to an Intranet network) from the unsecured side of the LAN/WAN network (typically the File Servers attached to the Internet network).

The network device controls and filters each file it receives, according to some anti-virus criteria. For that, it uses:

- predefined rules, and
- the virus-free Certificate associated with each file.

1 The network device therefore protects the Client Workstations
2 from viruses. This network device is called Virus-free
3 Certificate Firewall (also referred to as VCF) (105).

4 **Virus-free Certificate Firewall**

5 Depending on some predefined rules, a Virus-free Certificate
6 Firewall (VCF) may have to add a Virus-free Certificate to a
7 file it receives from a File Server. In this case, the
8 Virus-free Certificate Firewall retrieves the virus-free
9 Certificate from a Virus-free Certificate Cache (VCC) (106).
10 If the virus-free Certificate cannot be retrieved from a
11 Virus-free Certificate Cache, the Virus-free Certificate
12 Firewall retrieves the Virus-free Certificate from a
13 Virus-free Certificate Proxy (107).

14 **Virus-free Certificate Cache**

15 A Virus-free Certificate Cache (VCC) (106) is used to store
16 existing Virus-free Certificates within the LAN/WAN network.
17 These Virus-free Certificates can be retrieved by systems
18 attached to the LAN/WAN network (typically the Virus-free
19 Certificate Firewall). Retrieving an existing Virus-free
20 Certificate is a more efficient operation (in term of
21 performance) than building a new Virus-free Certificate.

22 Multiple Virus-free Certificate Caches can be attached to
23 the LAN/WAN network. In this case, the multiple Virus-free
24 Certificate Caches communicate across the LAN/WAN network in
25 order to exchange the Virus-free Certificates.

26 **Virus-free Certificate Proxy**

27 A Virus-free Certificate Proxy (VCP) (107) acts as a single
28 Virus-free Certificate Authority (VCA) within the LAN/WAN

1 network in order to provide virus-free Certificates to any
2 system attached to the LAN/WAN network. When multiple
3 Virus-free Certificate Authorities (VCAs) are available (for
4 instance, one VCA per software vendor), the Virus-free
5 Certificate Proxy (VCP) identifies a Virus-free Certificate
6 Authority (VCA) which has authority for building a Virus-free
7 Certificate for of a particular file, and retrieves the
8 Virus-free Certificate from said identified Virus-free
9 Certificate Authority (VCA).

10 **Virus-free Certificate Retrieval**

11 Any system attached to the LAN/WAN network can retrieve a
12 virus-free Certificate (VC) for a specific file from either:

- 13 • A Virus-free Certificate Authority (VCA), which builds
14 the VC. The virus-free Certificate (VC) is said to be
15 "authoritative" because the Virus-free Certificate
16 Authority (VCA) has authority to build a valid VC, using
17 for instance the latest anti-virus programs and levels.
- 18 • A Virus-free Certificate Proxy (VCP), which relays the
19 request for virus-free Certificate (VC) to one or
20 multiple Virus-free Certificate Authorities (VCAs). Since
21 a Virus-free Certificate Proxy (VCP) dynamically
22 retrieves "authoritative" virus-free Certificates (VCs)
23 from VCAs, it also provides "authoritative" virus-free
24 Certificates (VCs).
- 25 • A Virus-free Certificate Cache (VCC). A VCC has not
26 authority to build virus-free Certificates (VCs) but it
27 maintains local copies of virus-free Certificates (VCs).

1 Possibly, a virus-free Certificate (VC) retrieved from a
 2 Virus-free Certificate Cache (VCC) may be invalid. For
 3 instance, the virus-free Certificate (VC) is expired
 4 since it was initially built by a Virus-free Certificate
 5 Authority (VCA). A Virus-free Certificate Cache (VCC)
 6 does not provide "authoritative" virus-free Certificates
 7 (VCs).

8 **Virus-free Certificate**

9 Figure 2 describes the content of a virus-free Certificate
 10 (VC) according to the present invention. The virus-free
 11 Certificate reuses the standard X.509 Certificate format. It
 12 comprises the signature of the file and therefore is bound to
 13 this file. The main difference between a X.509 Certificate and
 14 the virus-free Certificate is that the virus-free Certificate
 15 comprises:

- 16 • an anti-virus name and level;
- 17 • a signature of the file.

18 The virus-free Certificate (200) includes the following
 19 fields:

- 20 • **File name (201):** This is the "name" of the file protected
 21 that the virus-free Certificate protects.
- 22 • **Issuer (202):** This is the "name" of the third party that
 23 issued/generated the virus-free Certificate. This third
 24 party is the Virus-free Certificate Authority (VCA).
- 25 • **Public Key Value (203):** This is the public key of a
 26 public/private key pair. An associated field defines the
 27 public key algorithm that must be used to check the file

signature, for instance a RSA , Diffie-Hellman or DSA public key. The public key is provided by the Virus-free Certificate Authority. The corresponding private key is used by the VCA to build the signature of files. So the same private/public key pair may be used to build several virus-free Certificates from the same issuer (VCA). This public key within the virus-free Certificate is preferably used instead of the Virus-free Certificate Authority public key used to validate only the present certificate signature and not the file signature. A public key for decrypting the imbedded signature is added within the virus-free Certificate because the Virus-free Certificate Authority public key is generally longer and more complex. The validity of keys may also differ between the Virus-free Certificate Authority public key and the virus-free Certificate public key. Anyway, because the virus-free Certificate is signed by the Virus-free Certificate Authority, the use of the virus-free Certificate public key is secure.

- **Validity (204):** Two fields are used to define the period of validity (valid from date 1 and valid to date 2).
- **Serial Number (205):** This field provides a unique virus-free Certificate serial number for the issuer.
- **Certificate Signature (206):** The certificate signature is an encrypted digest generated by the Virus-free Certificate Authority (VCA) for authenticating the whole Certificate. The digest results from the hashing of the virus-free Certificate. The digest is encrypted using the Virus-free

Certificate Authority (VCA) private key. The certificate signature results from the encrypted digest and "certifies" that the file signature is encrypted by the private key associated with the virus-free certificate public key (203). The Virus-free Certificate Authority (VCA) public key is different from the virus-free Certificate public key and is either preloaded in the web browser or given by a trusted entity. The VCA public key is used to retrieve the original hashing of the full certificate. The Virus-free Certificate Authority (VCA) can use the same set of virus-free certificate private / public keys (203) for all the files generated during a given period of time so the cross-checking of the issuer authentication can be easily performed time to time, when a new set of keys is used. Once the virus-free Certificate public key for a issuer is validated, it can be reused for several files certified by the same issuer which reduces the number of virus-free Certificate public keys.

- **File Signature (207):** The File Signature is verified using the public key value given in the virus-free Certificate. The file signature (hashing) performed by the Virus-free Certificate Authority (VCA) on the file is encrypted using the VCA private key. Using the VCA public key allows to retrieve the original file signature.
- **Anti-virus Checker (208):** This field gives an indication of how the virus-free Certificate has verified that the file was virus-free. The Anti-virus Checker comprises the name and the level of the anti-virus program. Several anti-virus programs and levels may be appended to reinforce the efficiency of the anti-virus detection.

- 1 • **Certificate Structure (209):** This field describes the size
2 and the content of the virus-free Certificate fields. The
3 number or anti-virus program is defined in this field.
- 4 • If the virus-free Certificate uses a standard format
5 (minimum size of a virus-free Certificate), this field is
6 optional.
 - 7 • If the size of the virus-free Certificate is above the
8 size of the standard format (above the minimum size),
9 this field is mandatory and defines the size of the
10 fields comprised in the virus-free Certificate.

11 **Virus-free Certificate Rules Table**

12 Figure 3 describes the table used by the Virus-free
13 Certificate Firewall (VCF) (105). Said table provides some
14 indications for controlling and filtering the files received
15 by the Virus-free Certificate Firewall (VCF). This table is
16 called Virus-free Certificate (VC) Rules Table (301).
17 The Virus-free Certificate Table (301) (a flat file in a
18 preferred embodiment) is typically created by the Network
19 Administrator in charge of the LAN/WAN network. This table
20 associates with each file some anti-virus criteria. Said
21 anti-virus criteria are used by the Virus-free Certificate
22 Firewall (VCF) for controlling and filtering of each file.

23 Each file is identified by its characteristics (303), which
24 can be one or a combination of the following information:

- 25 • the source system which has originated the file,
- 26 • the file name,
- 27 • the file type,

- 1 • optionally, the file size,
2 • optionally, the file CRC (a file signature).

3 The anti-virus criteria (307) associated with each file
4 comprise:

- 5 • an information indicating whether or not a Virus-free
6 Certificate is required for this file,
7 • an information indicating a list of information comprised
8 within a virus-free Certificate (VC) and that must be
9 check to determine that the virus-free Certificate (VC)
10 is valid,
11 • an information indicating whether or not the file must be
12 discarded, when a virus-free Certificate (VC) is required
13 for this file, and when either:
14 • no Virus-free Certificate is associated with the file,
15 or
16 • the Virus-free Certificate which is associated with
17 the file, is not valid.
18 • a list of anti-virus programs and levels that must be
19 used for building the Virus-free Certificate associated
20 with the received file. Typically, this list of
21 anti-virus programs and levels is used by the authority
22 (a Virus-free Certificate Authority) which builds the
23 virus-free Certificate (VC).

24 The Virus-free Certificate Rules Table (301) comprises a list
25 of records (302). In order to minimize the number of records
26 in the table, files having the same characteristics are
27 grouped within file templates (303). The anti-virus criteria
28 comprised within one record, apply to all files within the

1 file template. There is one record for each file template,
2 each record comprising the following information:

3 • (303) a **File Template** section, which comprises the
4 following information (fields in the record):

5 • (304) **File_Source**: This is the identifier of the source
6 system or the plurality of source systems that have
7 originated files. For instance, the File_Source can be
8 the IP address of the File Server which has originated
9 all files within a specific software company (for
10 instance the IP address of www.ibm.com File Server).

11 • (305) **File_Name**. This is the name of one or multiple
12 files. The File_Name can be either:

13 • an explicit file name which identifies a unique file.
14 For instance a File_Name with the value "setup1"
15 identifies the file called "setup1".

16 • a wildcarded file name which identifies multiple
17 files. For instance, File_Name with the value "setup*"
18 identifies any file which name starts with "setup"
19 (for instance "setup1", "setup_ok",...).

20 • (306) **File_Type**. This is the type of the file (for
21 instance "exe", "doc", "avi", "html",...). A File_Type
22 identifies any file of this type. For instance, a
23 File_Type with the value "exe" identifies all files which
24 type is "exe" (for instance "start.exe", "word.exe",...).

25 Optionally, the File Template section can also comprise a
26 file size. The file size is then the size of all files
27 within the same File Template. The file size can possibly

1 used to differentiate files having the same File_Source,
2 File_Name, and File_Type.

3 Optionally, the File Template section can also comprise a
4 file CRC (a file signature). The file CRC is then the
5 signature of the file identified by the File Template. The
6 file CRC can possibly used to differentiate files having
7 the same File_Source, File_Name, and File_Type.

8 • (307) a **File Anti-virus Criteria** section, which comprises
9 the following information (fields in the record):

10 • (308) **VC_Required**. This information indicates whether or
11 not a Virus-free Certificate must be associated with any
12 file within the file template. The possible values of
13 VC_Required are "yes" and "no".

14 • (309) **Valid_VC_Required**. This information indicates the
15 list of information comprised within a virus-free
16 Certificate (VC) and which must be validated by the
17 Virus-free Certificate Firewall (VCF) in order to
18 determine that the virus-free Certificate (VC) is valid.
19 For instance, a virus-free Certificate (VC) with an
20 expired date may be considered as valid. This information
21 applies to any file within the file template. By default,
22 all information comprised within a virus-free Certificate
23 (VC) must be checked and validated by the Virus-free
24 Certificate Firewall (VCF) in order to determine that the
25 virus-free Certificate (VC) is valid.

- (310) **File_Discard**. This information indicates whether or not the file must be discarded, when "VC_Required" is set to "yes", and when either:
 - no Virus-free Certificate is associated with the file, or
 - the Virus-free Certificate which is associated with the file, is not valid.
 The possible values are "yes" and "no".

- (311) **Required_AV_Checker_List**. This is the list of anti-virus programs and levels which must be used for the anti-virus processing of any file within the file template. Said list is therefore the list of anti-virus programs and levels that must be used by the authority (typically a VCA) which builds the Virus-free Certificate associated with any file within the file template.

A default record is defined in the Virus-Free Certificate Rules Table. This default record comprises anti-virus criteria that must be used by the Virus-free Certificate Firewall (VCF) for processing files which are not explicitly listed in a specific record.

Virus-free Certificate Firewall

According to the present invention, a purpose of a Virus-free Certificate Firewall (VCF) is to control and filter each file it receives, according to some anti-virus criteria and using Virus-free Certificates. For each file it receives, the Virus-free Certificate Firewall (VCF) performs the following operations:

- 1 • retrieving the anti-virus criteria for said received
- 2 file, from the Virus-free Certificate Rules Table
- 3 (301),
- 4 • checking whether or not said received file is
- 5 associated with a Virus-free Certificate,
- 6 • checking whether or not the Virus-free Certificate
- 7 which is associated with said received file, is valid,
- 8 based on said anti-virus criteria,
- 9 • when required by said anti-virus criteria, retrieving
- 10 a valid Virus-free Certificate associated with said
- 11 file. Typically, said Virus-free Certificate is
- 12 retrieved either from a Virus-free Certificate
- 13 Authority (VCA), or from Virus-free Certificate Cache
- 14 (VCC) or from a Virus-free Certificate Proxy (VCP),
- 15 • discarding the file, when the anti-virus criteria are
- 16 not satisfied.
- 17 • forwarding the file, possibly along with its
- 18 associated valid Virus-free Certificate, when the
- 19 anti-virus criteria are satisfied.

20 Typically, the Virus-free Certificate Firewall (VCF) is either

21 a dedicated network device attached to the LAN/WAN network, or

22 an existing network device (for instance an existing Firewall)

23 which is enhanced to provide the Virus-free Certificate

24 Firewall (VCF) functions.

25 Figure 4 is a flow chart which refers to the internal logic of

26 the Virus-free Certificate Firewall (VCF). The VCF:

- 27 • (401): receives a file

- 1 • (402): retrieves the anti-virus criteria associated with
2 the file, from the Virus-free Certificate Rules Table
3 (403). Said file anti-virus criteria are comprised in the
4 VC Rules Table record which is associated with the received
5 file. Said record is identified using its "File_Source",
6 "File_Name", and "File_Type" fields, and using information
7 retrieved from the received file (its name, its type, and
8 its source which is typically identified by the IP address
9 of the system which originated the file). Typically, the
10 anti-virus criteria are comprised in the fields
11 ("VC_Required", "Valid_VC_Required", "File_Discard", and
12 "Required_AV_Checker_List") of the file anti-virus
13 criteria section of the identified record.
- 14 • (404): tests whether or not a Virus-free Certificate (VC)
15 is associated with the received file. Preferably, the
16 virus-free Certificate (VC) has been received within the
17 file or through separate means.
- 18 • If a virus-free Certificate (VC) is found:
 - 19 • (405) tests whether or not the received virus-free
20 Certificate (VC) is valid. Typically, and by default, the
21 VC is valid when all the following conditions are
22 satisfied:
 - 23 • The VCA which has issued the VC, and identified by
24 the "issuer" field of the VC, is a trusted VCA.
25 Typically, this is done by checking that the VCA is
26 comprised within a list of trusted VCAs This list is
27 configured on the VCF by a Network Administrator.
 - 28 • The VC is authenticated using:

- 1 • its "certificate signature" field and
- 2 • the public key of the VCA which has issued the
- 3 VC.
- 4 Typically, the VCF retrieves said public key from a
- 5 list of trusted VCA certificates which have been
- 6 placed on the VCF by a Network Administrator, or
- 7 which have been retrieved by the VCF through secure
- 8 means.
- 9 • The VC date is correct. Typically, the current date
- 10 is comprised in the date interval provided in the
- 11 "validity" field.
- 12 • The VC has not been revoked. Typically, this is done
- 13 using :
- 14 • the "issuer" and the "serial number" fields of
- 15 the VC, and
- 16 • a certificate revocation list (CRL) issued by the
- 17 VCA.

18 Possibly, the "Valid_VC_Required" indication retrieved
 19 from the VC Rules Tables in step (402), indicates the
 20 list of the above conditions that must be satisfied. For
 21 instance, "Valid_VC_Required" may indicate that a VC with
 22 an expired date is considered as valid.

- 23 • If the virus-free Certificate (VC) is valid:
- 24 • (406) tests whether or not the anti-virus processing
- 25 of the file is OK. This anti-virus processing consists
- 26 in checking that the VCA which issued the VC has
- 27 tested the file for viruses. This checking uses at

1 least the list of required anti-virus programs and
2 levels comprised in the file anti-virus criteria.

- 3 • The list (List1) of anti-virus programs and levels
4 used by the VCA to test the file for viruses is
5 retrieved from the "anti-virus checker" fields of
6 the VC.
- 7 • The list (List2) of anti-virus programs and levels
8 that must be used to test the file for viruses, is
9 retrieved from the "Required_AV_Checker_List" field
10 of the file anti-virus criteria section of the
11 selected record (said selected record has been
12 retrieved in (402) from the VC Rules table).

13 By default, the anti-virus processing is OK when List2
14 is a subset of List1.

15 Possibly, the "Valid_VC_Required" indication which has
16 been retrieved from the VC Rules Tables in step (402),
17 indicates if the anti-virus processing is OK when:

- 18 • Both List1 and List2 are identical, or
- 19 • List2 is a subset of List1 (this is the
20 default).

- 21 • If the anti-virus processing is OK:

- 22 • (407) tests whether or not the file signature is OK.
23 This test compares the file signature calculated by
24 the VCF and the file signature comprised within the
25 VC (in the "file signature" field). If both are the
26 same, then the file signature is OK, which means
27 that the received file has not been modified since
28 it was checked for viruses by the VCA.

- 1 • If the file signature is OK:
 - 2 • (408) updates a Virus-free Certificate (VC)
 - 3 Cache. An update request comprising the file VC,
 - 4 is sent to a VC Cache by the VCF. Said request
 - 5 typically comprises the file name and type, the
 - 6 file size, the file CRC, and the file Virus-free
 - 7 Certificate. Preferably, the address of a VC
 - 8 Cache is a predefined parameter of the VCF.
 - 9 This step is optional, and is obviously only done
 - 10 when a VC Cache has been defined within the
 - 11 LAN/WAN network.
- 12 • (409) forwards the received file, along with its
- 13 valid VC.
- 14 The VCF then waits for the next file to process
- 15 in (401).
- 16 • If the file signature is not OK:
 - 17 • (410) stores an information indicating that the
 - 18 file signature is not OK (or any other detected
 - 19 error).
 - 20 • (411) retrieves a VC from a Virus-free
 - 21 Certificate (VC) Proxy. The VCF sends a request
 - 22 to a VC Proxy, in order to retrieve a VC
 - 23 associated with the received file. Said VC is
 - 24 authoritative since it is built and issued by a
 - 25 VCA. Possibly, the VCF retrieves an authoritative
 - 26 VC directly from a VCA, when no VC Proxy is

defined within the LAN/WAN network. Said request typically comprises:

- The file name and type, the file size (optional), and the file CRC.
- Optionally, a reduced VC which contains the "issuer" information. Said "issuer" information identifies the preferred VCA, and provides the VCP with some selection criteria which can be used when multiple VCs are available for the same file (typically, one VC per VCA).
- Optionally, a list of anti-virus checkers, that should preferably be used by the VCA to build the VC.

Preferably, the address of the VC Proxy (or VCA) is a predefined parameter of the VCF.

- (412) tests whether or not a valid VC has been retrieved.

A VC may have been retrieved from the VCP (or VCA) in step 411. Said VC is tested to make sure it is valid and has not been corrupted for instance by a malicious system stealing the identity of a trusted VCA. The VC is validated using the same criteria and possibly the "Valid_VC_Required" indication, as in steps (405), (406), and (407). Typically, the test determines that:

- The VCA which has issued the VC is a trusted VCA.
- The VC is authenticated.
- The VC date is correct.

1 • The VC has not been revoked.
2 • The anti-virus processing is OK.
3 • The file signature is OK.

4 • If a valid VC has been retrieved:

5 • (408) updates a Virus-free Certificate (VC)
6 Cache.

7 • (409) forwards the received file, along with
8 its valid VC. The VCF then waits for the next
9 file to process in (401).

10 • If a valid VC has not been retrieved:

11 • (413) discards the file, and stores an
12 information which indicates the discard and
13 that a valid VC has not been retrieved. The VCF
14 then waits for the next file to process in
15 (401).

16 • If the anti-virus processing is not OK:

17 • (414) tests whether or not the VC has been
18 retrieved from a VC Cache. The information
19 indicating that the VC has been retrieved from a
20 VC Cache is set in step (420).
21
22 • If the VC has been retrieved from a VC Cache (in
23 step (420))

10207076260

- 1 • Continues in step (410), in order to possibly
- 2 retrieve an authoritative VC from a VC Proxy
- 3 (or VCA) .

- 4 • If the VC has not been retrieved from a VC Cache

- 5 • Continues in step (416), in order to possibly
- 6 retrieve a VC from a VC Cache.

- 7 • If the VC is not valid:

- 8 • Continues in step (414), in order to possibly retrieve
- 9 a valid VC, either from a VC Proxy (or VCA) or from a
- 10 VC Cache.

- 11 • If a VC is not found:

- 12 • (415) tests whether or not a VC is required for the
- 13 received file. The test uses the "VC_Required" indication
- 14 retrieved from the VC Rules Table in step (402).

- 15 • If a VC is required for the received file:

- 16 • (416) stores an information indicating an error
- 17 condition. Possibly, a file which should have a VC has
- 18 been received without a VC, or a file has been
- 19 received with an invalid VC.

- 20 • (417) tests whether or not the file must be discarded.
- 21 The test uses the "File_Discard" indication retrieved
- 22 from the VC Rules Table in step (402). The file must

RECEIVED

- 1 be discarded when "File_Discard" is set to "yes", and
2 when "VC_Required" is set to "yes", and when either:
3 • no VC is associated with the file, or
4 • the VC which is associated with the file, is not
5 valid.
- 6 • If the file must be discarded:
- 7 • (418) discards the file, and stores an information
8 which indicates the discard. The VCF then waits for
9 the next file to process in (401).
- 10 • If the file must not be discarded:
- 11 • (420) retrieves a VC from a Virus-free Certificate
12 (VC) Cache. The VCF sends a request to a VC Cache,
13 in order to retrieve a VC associated with the
14 received file. Said request typically comprises:
15 • The file name and type, the file size (optional),
16 and the file CRC.
17 • Optionally, a reduced VC which contains the
18 "issuer" information. Said "issuer" information
19 identifies the preferred VCA, and provides the
20 VCC with some selection criteria which can be
21 used when multiple VCs are available for the same
22 file (typically, one VC pr VCA).
23 • Optionally, a list of anti-virus checkers. Said
24 list indicates the list of anti-virus checkers
25 that should preferably be used to build the VC,
26 and provides the VCC with some selection criteria

1 which can be used when multiple VCs are available
2 for the same file (typically, one VC per VCA).

3 Preferably, the address of the VC Cache is a
4 predefined parameter of the VCF. This step is
5 optional, and is obviously only done when a VC
6 Cache has been defined within the VCF.

7 • (421) tests whether or not a VC has been retrieved
8 from a VC Cache:

9 • If a VC has been successfully retrieved from a VC
10 Cache:

11 • Continues in step (405) in order to validate the
12 retrieved VC. Since the VC has been retrieved
13 from a VC Cache, the VC is not authoritative and
14 must be validated. An indication that the VC has
15 been retrieved from a VC Cache is also stored for
16 internal use (in step (414)).

17 • If a VC has not been successfully retrieved from a
18 VC Cache:

19 • Continues in step (411) in order to retrieve an
20 authoritative VC from a VC Proxy (or from a VCA).
21

22 **Virus-free Certificate Cache Table**

23 Figure 5 describes the table used by the Virus-free
24 Certificate Cache (VCC) (106). This table, called Virus-free

1 Certificate Cache Table, is dynamically built by the VCC and
2 comprises a local copy of Virus-free Certificates which have
3 been transmitted through the LAN/WAN network. The table (501)
4 comprises for each file, one or multiple associated Virus-free
5 Certificates.

6 In the VC Cache Table (501), each file is identified by its
7 characteristics (503), which are:

- 8 • the file name,
- 9 • the file type,
- 10 • optionally, the file size,
- 11 • the file CRC.

12 In the VC Cache Table, each file is associated with one or
13 multiple VC sections (508), each VC section comprising the
14 following information:

- 15 • The VC associated with this file,
- 16 • The date of the latest request to retrieve said VC,
- 17 • The number of requests (hits) to retrieve said VC.

18 The table comprises a list of records (502). There is one
19 record for each file, each record comprising the following
20 information:

- 21 • (503) a **File** section comprising the following information
22 (fields in the record):

- 23 • (504) **File_Name**. This is the name of the file. Typically,
24 this is the explicit file name. For instance a File_Name
25 with the value "setup1" identifies the file called
26 "setup1".

FILE NO. 93-2626

- 1 • (505) **File_Type**. This is the type of the file (for
- 2 instance "exe", "doc", "avi", "html",...).
- 3 • (506) **File_Size**. This is the size of the file. This field
- 4 is optional.
- 5 • (507) **File_CRC**. This is the CRC (a file signature) of the
- 6 file.

7 The combination of File_Name and File_Type may identify in

8 a unique way one file.

9 If multiple files have the same name and type (for

10 instance, multiple "setup.exe" files originated from

11 multiple sources), each one of these said file may then be

12 differentiated in a unique way by its file size (for

13 instance the size of one "setup.exe" is 2 kilobytes, and

14 the size of another "setup.exe" is 4 kilobytes).

15 If multiple files have the same name, type, and size, each

16 one of these said file is then differentiated in a unique

17 way by its file CRC which is a file signature (hashing)

- 18 • (508) a **Virus-free Certificate (VC)** section comprising the
- 19 following information (fields in the record):

- 20 • (509) **Virus-free_Certificate (VC)**. This is the VC
- 21 associated with the file.

- 22 • (510) **Last_Date**. This is the date of the latest request
- 23 received by the VC Cache to retrieve or to update this
- 24 VC. Typically, this date is used when the VC Cache is
- 25 maintained and when for instance the oldest records have
- 26 to be deleted.

- 27 • (511) **Number_Hits**. This is the number of requests (hits)
- 28 that have been received by the VC Cache to retrieve this
- 29 VC. Typically, this number of hits is used when the VC

1 Cache is maintained and when for instance the records
2 with the lowest number of hits have to be deleted.

3 Possibly, multiple VC sections are associated with one file
4 within the same record to support multiple VCs per file. This
5 is for instance necessary when one file has one VC per VC
6 Authority (VCA), each VC Authority using for instance one
7 specific anti-virus program. In this case, each record is
8 still identified by its File section, but multiple VC sections
9 are then associated with said File section (one VC section per
10 VC).

11 **Virus-free Certificate Cache**

12 According to the present invention, a Virus-free Certificate
13 Cache (VCC) (106) stores existing Virus-free Certificates
14 within the LAN/WAN network. Said existing Virus-free
15 Certificates can then be retrieved by systems attached to the
16 LAN/WAN network (typically the Virus-free Certificate
17 Firewall). Retrieving an existing Virus-free Certificate is
18 more efficient than building a new Virus-free Certificate, and
19 therefore provides better performance.
20 Multiple Virus-free Certificate Caches can be attached to the
21 LAN/WAN network. In this case, the multiple VCCs communicate
22 across the LAN/WAN network in order to exchange the Virus-free
23 Certificates.

24 The VC Cache performs the following functions:

- 25 • Storing copies for each file of the Virus-free
26 Certificate (VC) or of the multiple Virus-free
27 Certificates in a VC Cache Table.

- 1 • Processing each request for updating the VC Cache
- 2 Table with a new Virus-free Certificate. This
- 3 operation comprises the steps of:
- 4 • Dynamically updating a local VC Cache Table with
- 5 a copy of said new Virus-free Certificate.
- 6 • Possibly updating one or multiple remote VC Cache
- 7 Tables with said new Virus-free Certificate.
- 8 • Processing each request for retrieving a VC associated
- 9 with a specific file. This operation comprises the
- 10 steps of:
- 11 • Preferably retrieving a VC from a local VC Cache
- 12 Table.
- 13 • Possibly retrieving a VC from a remote VC Cache
- 14 Table.
- 15 • Maintaining the VC Cache Table, by discarding from the
- 16 VC Cache Table the invalid VCs.
- 17 • Possibly pre-loading the VC Cache Table with one or
- 18 multiple VCs which are directly retrieved from one or
- 19 multiple VC Authorities or VC Proxies.

20 Furthermore, the operation of preferably retrieving a VC from
21 a local VC Cache Table, comprises the following steps:

- 22 • Identifying a preferred VC in the VC Cache Table, using
- 23 one or multiple selection criteria, said selection
- 24 criteria comprising:
- 25 • the name of said specific file,
- 26 • the type of said specific file,
- 27 • optionally the size of said specific file,
- 28 • the CRC of said specific file,

- 1 • optionally a preferred list of anti-virus programs and
2 levels,
3 • optionally the identifier of a preferred VCA.
4 • Retrieving said preferred VC from the VC Cache Table.

5 Typically, the VCC is either a dedicated network device
6 attached to the LAN/WAN network, or an existing network device
7 (for instance an existing WEB Proxy system) which is enhanced
8 to provide the VCC functions.

9 **Virus-free Certificate Cache Table Maintenance**

10 The VCC periodically maintains the VC Cache Table. This
11 operation comprises the following steps:

- 12 • Checking the validity of each VC in the VC Cache Table
13 using the VC fields. Typically, this checking uses the
14 "validity" and the "serial number" fields of the VC to
15 determine if the VC has expired or if the VC has been
16 revoked.
17 • Discarding from the local VC Cache Table:
18 • each VC section with an invalid VC,
19 • Each record with no VC section.

20 The VC Cache Table maintenance is preferably performed at
21 regular time intervals, for instance every night. The
22 maintenance process can be triggered either automatically or
23 manually. The periodicity of said maintenance is for instance
24 a predefined parameter of the VCC.

25 Optionally, the VC Cache can discard in one or multiple remote
26 VC Cache Tables, records comprising a VC which is no longer
27 valid. This operation is optional since preferably each VCC

1 maintains its own local VC Cache Table, in order to avoid
2 unnecessary traffic across the LAN/WAN network.

3 **Virus-free Certificate Cache Table Pre-Loading**

4 The VC Cache can pre-loads its local VC Cache Table with one
5 or multiple VCs directly retrieved from one or multiple VC
6 Authorities (VCAs) or VC Proxies (VCPs). In this case, the VC
7 Cache is configured (typically by a Network Administrator)
8 with a list of VCAs (or VCPs). The VC Cache periodically
9 retrieves VCs from the VCAs within said list, and populates
10 the VC Cache Table with records comprising said retrieved VCs.

11

12 Typically, the VC Cache can retrieve from a VCA one file (for
13 instance using FTP) comprising the latest VCs created by said
14 VCA. The VC Cache then creates one record in the VC Cache
15 Table, for each VC comprised within said retrieved file.
16 The pre-loading of the VC Cache Table is preferably performed
17 at regular time intervals, for instance every night. The
18 pre-loading process can be triggered either automatically or
19 manually. The periodicity of said pre-loading is for instance
20 a predefined parameter of the VCC.

21 **Virus-free Certificate Cache: Virus Free Certificate Retrieval**

22

23 Figure 6 is a flow chart which refers to the internal logic of
24 the Virus-free Certificate Cache (VCC) and more particularly
25 to the method for processing a request for retrieving a VC for
26 a particular file. The Virus-free Certificate Cache (VCC):

- 27 • (601): receives a request to retrieve a VC for a particular
28 file. Typically, said request for VC typically comprises:

- 1 • The name, and type of the file,
- 2 • The size of the file (optionally),
- 3 • The CRC of the file. This is a file signature identifying
- 4 the file.
- 5 • Optionally, the list of anti-virus programs and levels
- 6 (referred to as "RCV_AV_Checker_List"), that should
- 7 preferably be comprised within the list of anti-virus
- 8 programs and levels which have been used to build the
- 9 requested VC. Typically, this information is used by the
- 10 VCC to select the preferred VC, when multiple VCs are
- 11 available for the same file.
- 12 • Possibly, the list is empty, which means that any
- 13 anti-virus program and level can be used.
- 14 • Optionally, a reduced VC comprising only the "issuer"
- 15 field (referred to as "RCV_issuer"), and identifying the
- 16 preferred VCA. The reduced VC then provides the VCC with
- 17 some selection criteria. These selection criteria can be
- 18 used when multiple VCs are available for the same file
- 19 (typically, one VC available per VCA).
- 20 • (602) retrieves one VC from the VC Cache Table (603).
- 21 Typically, this operation comprises the following steps:
- 22 • Selecting one record (called "Record_S") within the VC
- 23 Cache Table, using:
- 24 • Its File_Name and File_Type, which must be identical
- 25 to the file name and file type comprised in the
- 26 received request.
- 27 • If multiple records (files) have the same File_Name
- 28 and File_Type (for instance, multiple "setup.exe"
- 29 files originated from multiple sources), each record
- 30 (file) can then be differentiated in a unique way by

- 1 using the File_Size (for instance the size of one
- 2 "setup.exe" is 2 kilobytes, and the size of another
- 3 "setup.exe" is 4 kilobytes).
- 4 • In the unlikely event of multiple records (files)
- 5 having the same File_Name, File_Type, and File_Size,
- 6 each record (file) is then differentiated in a unique
- 7 way by using the File_CRC.
- 8 • Selecting one VC within said selected record:
- 9 • When only one VC section is comprised within
- 10 "Record_S", the selected VC is the VC comprised within
- 11 said VC section.
- 12 • When multiple VC sections are comprised within
- 13 "Record_S", each one comprising one VC, the selected
- 14 VC is preferably identified using:
- 15 • The "issuer" field of the VC. This issuer field is
- 16 equal (or comprises) the "RCV_issuer" possibly
- 17 received in the request.
- 18 • The "Anti-virus Checker" fields of the VC. These
- 19 Anti-virus Checkers are equal (or comprise) the
- 20 "RCV_AV_Checker_List" possibly received in the
- 21 request.
- 22 • (604) tests whether or not a VC has successfully been
- 23 retrieved from the VC Cache Table.
- 24 • If a VC has been retrieved from the VC Cache Table
- 25 • (605) updates in the VC Cache Table (603), the
- 26 "Last_Date" and "Number_Hits" fields of the VC section
- 27 comprising the VC.
- 28 • Updates "Last_Date" with the current date

- 1 • Increments "Number_Hits".
- 2 • (606) answers the received request for VC, with a
- 3 positive response comprising the retrieved VC.
- 4 Optionally, the VC is validated before being sent back in
- 5 the response. The VCC then waits for the next request for
- 6 VC.
- 7 • If a VC has not been retrieved from the VC Cache Table
- 8 • (607) retrieves the VC from a remote VC Cache. The VCC
- 9 forwards the received request to one or multiple VC
- 10 Caches. The address of said VC Caches is typically a
- 11 predefined parameter of the VCC.
- 12 • (608) tests whether or not a VC has been retrieved from a
- 13 remote VC Cache.
- 14
- 15 • If a VC has successfully been retrieved from a remote VC
- 16 Cache:
- 17 • (609) stores said retrieved VC in the VC Cache Table.
- 18 Either a new record is created in the VC Cache Table,
- 19 or a new VC section is created in an existing record.
- 20 • (606) answers the received request for VC, with a
- 21 positive response comprising the VC which has been
- 22 retrieved. Optionally, the VC is validated before
- 23 being sent back in the response. The VCC then waits
- 24 for the next request for VC.

- 1 • If a VC has not been retrieved from a remote VC Cache:
- 2 • (610) answers the received request for VC, with a
- 3 negative response since no VC has been retrieved. The
- 4 VCC then waits for the next request for VC.

5 Virus-free Certificate Cache Table Update

6 Figure 7 is a flow chart which refers to the internal logic of

7 the Virus-free Certificate Cache (VCC) and more particularly

8 to a method for processing a request to update the VC Cache

9 Table. The Virus-free Certificate Cache Table (VCC):

- 10 • (701): receives a request to update the VC Cache Table,
- 11 with a VC associated with a particular file. Typically,
- 12 said request for VC typically comprises:
- 13 • The name, and the type of said file,
- 14 • The size of the file (optionally),
- 15 • The CRC of the file. This is a file signature which
- 16 identifies the file.
- 17 • The VC.
- 18 • (702) tests whether or not the received VC is valid.
- 19 The received VC is checked to be sure it is valid and has
- 20 not been corrupted for instance by a malicious system
- 21 sending corrupted VCs. Typically, the following VC
- 22 validation is processed:
- 23 • The VC is authenticated using its "certificate signature"
- 24 file,
- 25 • The VC "validity" is correct,
- 26 • The VC has not been revoked.

FOIA b 7 - DEX

- 1 • If the received VC is valid:
- 2 • (704) tests whether or not the VC Cache Table is full:
- 3 • If the VC Cache Table is full:
- 4 • (705) deletes some records of the VC Cache Table
- 5 (706). Typically, the records which are deleted are
- 6 selected according to the "Last_Date" and
- 7 "Number_Hits" fields. For instance, the records having
- 8 the oldest "Last_Date" and the lowest "Number_Hits"
- 9 are deleted.
- 10 • (707) updates the VC Cache Table (706), with the
- 11 received VC. Either a new record is created in the VC
- 12 Cache Table, or a new VC section is created in an
- 13 existing record.
- 14 • (708) updates one or multiple remote VC Caches. The
- 15 VCC forwards the received request to one or multiple
- 16 VC Caches. The address of said VC Caches is typically
- 17 a predefined parameter of the VCC. This step is
- 18 obviously bypassed when no remote VC Cache is defined.
- 19 • If the VC Cache Table is not full:
- 20 • Continues in step (707) to update the VC Cache Table
- 21 with the received VC.
- 22 • If the received VC is not valid:

1 • (703) stores an information indicating that a request for
2 VC Cache update has been received with an invalid VC, and
3 discards the request.

4 **Virus-free Certificate Proxy**

5 The Virus-free Certificate Proxy (VCP) (107) acts as a single
6 Virus-free Certificate Authority (VCA) within the LAN/WAN
7 network in order to provide Virus-free Certificates to any
8 system attached to the LAN/WAN network. When multiple VCAs are
9 available (for instance, one VCA per software provider), the
10 VCP identifies a VCA which has authority to build a Virus-free
11 Certificate (VC) for a particular file, and retrieves said
12 Virus-free Certificate (VC) from said identified VCA.

13 A VC Proxy is a method and system for retrieving a VC for a
14 particular file, from one or multiple VCAs. A VC Proxy
15 processes requests for VCs (a request for VC is a request to
16 retrieve one VC for a particular file). The method of
17 processing each received request for VC comprises the steps
18 of:

- 19 • Identifying a VCA which has authority to build a VC for
20 said particular file,
- 21 • Retrieving said VC from said identified VCA,
- 22 • When said requested VC has been retrieved, answering the
23 received request for VC, with a positive response
24 comprising said VC.

25 Furthermore, the step of identifying a VCA which has authority
26 to build a VC for a particular file, comprises the further
27 steps of:

- 1 • Retrieving the identifier of the VCA, using information
- 2 comprised within the received request for VC, or
- 3 • Identifying the VCA using
- 4 • information comprised within the received request for
- 5 VC, and
- 6 • information retrieved from a configured VC Proxy
- 7 Table.
- 8 • If no VCA is identified:
- 9 • Identifying a Virus-free Certificate Relay (VCR)
- 10 system, which can provide the identifier of a VCA.
- 11 • Retrieving from said identified Virus-free Certificate
- 12 Relay (VCR) system, the identifier of a VCA which has
- 13 authority to build said VC.
- 14 • If no VCR system is identified,
- 15 • identifying a configured default VCA using
- 16 information retrieved from a configured VC Proxy
- 17 Table.

18 Furthermore, the step of identifying a Virus-free Certificate
19 Relay (VCR) system, which can provide the identifier of a VCA
20 which has authority to build said VC, comprises the further
21 steps of:

- 22 • Retrieving the identifier of a Virus-free Certificate
- 23 Relay (VCR), from information comprised within said
- 24 received request for VC, or
- 25 • Identifying a Virus-free Certificate Relay (VCR) system
- 26 using information comprised within said received request
- 27 for VC, and using information retrieved from a configured
- 28 VC Proxy Table.

1 Furthermore, the step of retrieving said VC for said specific
2 file from said identified VCA, consists in either:

- 3 • Retrieving the VC, using a short request sent to said
4 identified VCA and which does not comprise said specific
5 file; or
- 6 • Retrieving the VC, using a full request sent to said
7 identified VCA and which comprises said specific file.

8 Typically, the VCP is either a dedicated network device
9 attached to the LAN/WAN network, or an existing network device
10 (for instance an existing WEB Server) which is enhanced to
11 provide the VCP functions.

12 **Virus-free Certificate Proxy Environment**

13 Figure 8 depicts the logical anti-virus entities involved in a
14 VC Proxy environment. All the systems are attached to the
15 LAN/WAN network (802). The WAN may be for instance the
16 Internet network, or/and a company private Intranet network.
17 One or multiple Virus-free Certificate Authorities (VCAs)
18 (804) are attached to the LAN/WAN network (802). They have
19 authority to build Virus-free Certificates (VCs). Typically,
20 each VCA has authority to build the VCs associated with
21 specific files. There can be:

- 22 • one VCA for each software provider (companies such as
23 IBM, Symantec). For instance, IBM may provide and attach
24 one VCA to the Internet, in order to build VCs associated
25 with any file owned by IBM (for instance any software
26 package such as "Host On Demand"). For instance, the
27 Symantec company may attach one VCA to the Internet, in

1 A Virus-free Certificate Relay (VCR) (805) is therefore a
2 system which provides the identifier of one or multiple VCAs
3 having authority to build a VC for one or multiple files.
4 Typically, a VCR is a File Server (for instance an FTP Server)
5 which originates files owned by a software provider, and which
6 indicates the VCA that can be used to build a VC associated
7 with each file.
8 For instance, a small software provider may indicate on its
9 VCR (typically a FTP server) that the VCs associated with its
10 software packages can be built by one specific VCA (for
11 instance VeriSign). Any system requiring a VC for one of these
12 software packages can then contact the indicated VCA to
13 retrieve said VC.

14 **Virus-free Certificate Proxy Table**

15 Figure 9 describes the table used by the Virus-free
16 Certificate Proxy (VCP) (801). Said table (901) (a flat
17 file in a preferred embodiment) is preferably created by the
18 Network Administrator in charge of the LAN/WAN network. The
19 Virus-free Certificate Proxy Table (901):

- 20 • associates each file provider (for instance any software
21 provider such as IBM), with:
 - 22 • the identifier of one or multiple VCAs having
23 authority to build one or multiple VCs associated with
24 one or multiple files originated by said file
25 provider.
 - 26 • the identifier of one or multiple VCRs capable of
27 providing the identifier of one or multiple VCAs
28 having authority to build one or multiple VCs

1 The table comprises a list of records (902). There is one
2 record for each file provider, and optionally one record for
3 each file (or group of files), each record comprising the
4 following information:

5 • (903) a **File Provider** section, which comprises the
6 following information (fields in the record):

7 • (904) **File_Provider**. This is the identifier of the
8 software provider which originates one or multiple files.
9 Typically, said identifier is the company name of the
10 software provider (for instance IBM Corp.).

11 • (905) an optional **File Template** section, which identifies
12 one or multiple files having the same characteristics. Said
13 characteristics comprise the following information (fields
14 in the record):

15 • (906) **File_Identity**. This is the identifier of one or
16 multiple files within the file template, which has been
17 originally created by the software provider when it has
18 created said one or multiple files. Typically, said
19 identifier is imbedded within each file by the file
20 provider when the file is created. Said identifier
21 typically comprises the original file name, version, and
22 language.

23 • (907) **File_Name**. This is the name of one or multiple
24 files. The File_Name can be either:

25 • an explicit file name which identifies a unique file.
26 For instance a File_Name with the value "setup1"
27 identifies the file called "setup1".

- 1 • a wildcarded file name which identifies multiple
2 files. For instance, File_Name with the value "setup*" identifies any file which name starts with "setup"
3 (for instance "setup1", "setup_ok",...).
- 4
- 5 • (908) **File_Type**. This is the type of the file (for
6 instance "exe", "doc", "avi", "html",...). A File_Type identifies any file of this type. For instance, a
7 File_Type with the value "exe" identifies all files which
8 type is "exe" (for instance "start.exe", "word.exe",...).
- 9
- 10 • (909) **File_Size**. This is the size of the file.
- 11 • (910) **File_CRC**. This is the CRC (a file signature) of the
12 file.

13 In most cases, the File_Identity field provides sufficient
14 information to identify in a unique way one file template.
15 File_Name, File_Type, File_Size, and File_CRC are therefore
16 optional in the File Template section.

17 When File_Identity however, does not identify in a unique
18 way a file template (or when the File_Identity cannot be
19 retrieved for a specific file), the combination of
20 File_Name and File_Type may identify in a unique way a
21 file.

22 If multiple files have the same name and type (for
23 instance, multiple "setup.exe" files originated from
24 multiple sources), each file may then be differentiated in
25 a unique way by its file size (for instance the size of one
26 "setup.exe" is 2 kilobytes, and the size of another
27 "setup.exe" is 4 kilobytes).

28 In the unlikely event of multiple files having the same
29 name, type, and size, each file is then differentiated in a
30 unique way by its file CRC which is the file signature.

- 1 • (911) a **Virus-free Certificate Authority (VCA)** section,
2 which comprises the following information (fields in the
3 record):
- 4 • (912) **VCA_Id**. This is the identifier of a VC Authority
5 associated with the File Provider (and optionally with
6 the file template), and which therefore has authority to
7 build one or multiple VCs for each file originated by
8 said File Provider. Typically, this is the IP address of
9 the VCA.
- 10 • (913) **VCR_Id**. This is the identifier of a VC Relay
11 associated with the File Provider (and optionally with
12 the file template), and which therefore can provide the
13 identifier of one or multiple VCAs having authority to
14 build a VC for one or multiple files originated by said
15 File Provider. Typically, this is the IP address of the
16 VCR.
- 17 • (914) **AV_Checker_List**. This is the list of anti-virus
18 programs and levels used by the VCA identified by VCA_Id
19 when it builds VCs. This field is typically empty by
20 default, which means that said list is unknown.

21 One record typically comprises one VCA section. However,
22 one record possibly comprises multiple VCA sections, if one
23 File Provider has multiple VCAs (for instance, one VCA per
24 anti-virus program).

25 The VC Proxy Table comprises one default record, which
26 comprises the identifier (VCA_Identifier) of a Default VCA.
27 Possibly, multiple Default VCAs can be defined.

1 **Internal Logic of the Virus-free Certificate Proxy**

2 Figures 10a and 10b are flow charts which refer to the
3 internal logic of the Virus-free Certificate Proxy (VCP). The
4 VCP:

- 5 • (1001): receives a request to retrieve a VC for a
6 particular file. Typically, said request for VC comprises:
7 • One particular file,
8 • The name, and the type of this file,
9 • A list of anti-virus programs and levels (referred to as
10 "RCV_AV_Checker_List"), that must be used to build the
11 VC. The list may be empty, which means that any
12 anti-virus program and level can be used.
13 • Optionally, a reduced VC which only comprises the
14 "issuer" field (referred to as "RCV_issuer"), and which
15 identifies the preferred VCA. Said reduced VC provides
16 the VCP with some selection criteria which can be used
17 when multiple VCs are available for the same file
18 (typically, one VC available per VCA).
19 • (1002) tests whether or not the identifier of a VCA that
20 can be used to retrieve a VC, is comprised within the
21 received request for VC. Said identifier is then referred
22 to as "VCA_S_Id". Typically, said VCA identifier may be
23 imbedded within the received file (for instance in the
24 organization" section), or possibly derived from any
25 information retrieved from the received request.
26 • If VCA_S_Id is retrieved from information comprised within
27 the received request:

1 • (1003) retrieves one VC associated with the received file
2 from the VCA identified by VCA_S_Id, using a short
3 request. The VCP sends a short request to the VCA
4 identified by VCA_S_Id, in order to retrieve a VC
5 associated with the received file. Said short request
6 does not comprise the received file, in order to minimize
7 the traffic across the LAN/WAN network. Typically, said
8 short request for VC comprises the following information:
9 • The file name, and file type of the received file.
10 • The size of the received file,
11 • The CRC of the received file. Typically, the file CRC
12 is calculated using the received file.
13 • The list of anti-virus programs and levels that must
14 be used by the VCA to build the VC. This list may be
15 empty, which means that any anti-virus program and
16 level can be used.
17 • Typically, this list is the list which was received in
18 the request ("RCV_AV_Checker_List"). Possibly, this
19 list can also be a combination of
20 "RCV_AV_Checker_List" and some configuration
21 information of the VCP (for instance a Network
22 Administrator may want to enforce a specific and
23 updated list of anti-virus programs and levels that
24 must be used to build the VC).
25 Typically, the VCA_S_Id comprises the address (for
26 instance the IP address) of the VCA. Possibly, when no
27 specific VCA can be identified in a unique way, said
28 short request can be sent (broadcast) to all available
29 VCAs.

TOP SECRET

- 1 • (1004) tests whether or not a VC has been successfully
2 retrieved from the VCA:

- 3 • If a VC has been successfully retrieved from the VCA:

- 4 • (1005) answers the received request for VC, with a
5 positive response comprising the VC which has been
6 retrieved. Optionally, the VC is validated before
7 being sent back in the response. The VCP then waits
8 for the next request for VC.

- 9 • If a VC has not been successfully retrieved from the VCA:

- 10 • (1006) tests whether or not the VCA requires the
11 received file to build the VC. This test uses
12 information retrieved from the message which has been
13 received from the VCA in response to the short request
14 sent in (1003). Said information indicates whether or
15 not the VCA requires the file to build the VC.

- 16 • If the VCA requires the file to build the VC:

- 17 • (1007) retrieves a VC associated with the received
18 file from the VCA identified by VCA_S_Id, using a
19 full request.
20 The VCP sends a full request to the VCA identified
21 by VCA_S_Id, in order to retrieve a VC associated
22 with the received file. This full request comprises
23 the received file.

TOP SECRET

- 1 • (1008) tests whether or not a VC has been
2 successfully retrieved from the VCA:

- 3 • If a VC has been successfully retrieved from the
4 VCA:

- 5 • (1005) answers the received request for VC, with
6 a positive response comprising the VC which has
7 been retrieved. Optionally, the VC is validated
8 before it is sent back in the response. The VCP
9 then waits for the next request for VC.

- 10 • If a VC has not been successfully retrieved from the
11 VCA:

- 12 • (1009) tests whether or not the VCA is a Default
13 VCA, and when the VCA is a Default VCA, whether
14 or not another Default VCA is available. This
15 test preferably uses an indication which is
16 stored by the VCP when it identifies the VCA, and
17 which is determined using the VCP Table (1020).

- 18 • If the VCA is a Default VCA, and no other Default
19 VCA is available:

- 20 • (1010) stores an information indicating that
21 the VCP has not been able to retrieve a VC
22 associated with the particular file comprised
23 within the received request for VC. Possibly, a
24 notification is also sent to a Network
25 Administrator.

1 • (1011) answers the received request for VC,
2 with a negative response which indicates that
3 no VC has been retrieved. The VCP then waits
4 for the next request for VC.

5 • If the VCA is not a Default VCA, or if the VCA is
6 a Default VCA but another Default VCA is
7 available:

8 • (1012) excludes the VCA identified by VCA_S_Id,
9 from the list of VCAs which can possibly be
10 used to retrieve a VC for the received request.
11 Goes back to the identification of a VCA, in
12 step (1002), since another VCA (typically a
13 Default VCA) may be able to build the VC.

14 • If the VCA does not requires the file to build the VC:
15 The VCA is therefore not able to build the VC.

16 • Continues in step (1009) to test if the VCA is a
17 Default VCA.

18 • If VCA_S_Id is not retrieved from information comprised
19 within the received request:

20 • (1013) tests whether or not the identifier of a VC Relay
21 (VCR) is comprised within the received request for VC.
22 Said identifier is then referred to as "VCR_S_Id".
23 Typically, said VCR identifier may be imbedded within the
24 received file (for instance in the organization"

section), or possibly derived from any information
retrieved from the received request.

- If VCR_S_Id is retrieved from information comprised
within the received request:

- (1014) retrieves the identifier (VCA_S_Id) of a VCA
which has authority to build a VC for the received
file, from the VCR identified by VCR_S_Id. The VCP
sends a short request to the VCR identified by
VCR_S_Id, in order to retrieve the identifier
(VCA_S_Id) of a VCA which has authority to build a VC
for the received file. Typically, said short request
comprises the following information:

- The file name, and file type of the received
file.
- The size of the received file,
- The CRC of the received file. Typically, the file
CRC is calculated using the received file.
- The list of anti-virus programs and levels that
must be used by the VCA to build the VC.
Possibly, said list is empty, which means that
any anti-virus program and level can be used.
Typically, said list is the list which was
received in the request ("RCV_AV_Checker_List").
Possibly, said list can also be a combination of
"RCV_AV_Checker_List" and some configuration
information of the VCP (for instance a Network
Administrator may want to enforce a specific and
updated list of anti-virus programs and levels
that must be used to build the VC). Typically,

1 • the file provider identifier which is possibly
2 imbedded in the file.

3 Possibly, said record can also be identified by:

- 4 • Its "File_Identity" field, which must be equal
5 (or must comprise) the file identifier retrieved
6 from the file.
- 7 • Optionally, its "File_Name", "File_Type",
8 "File_Size" and "File_CRC" which must be equal to
9 (respectively) the name, type, size, and CRC of
10 received file.

11 At least a Default record can be identified.

12 • (1018) tests whether or not the identifier of a VCA
13 that can be retrieved from the VCP Table. A VCA
14 section is primarily selected within "Record_S".
15 Preferably, the selected VCA section is a section
16 which "AV_Checker_List" is equal to (or comprises)
17 "RCV_AV_Checker_List". The identifier of a VCA can be
18 retrieved from the VCP Table, when said selected VCA
19 section comprises a VCA Identifier in its "VCA_Id"
20 field.

21 • If the identifier (called VCA_S_Id) of a VCA can be
22 retrieved from the VCP Table:

23 • Continues in step (1003) in order to retrieve a VC
24 for the received file, from said identified VCA.

25 • If the identifier of a VCA cannot not be retrieved
26 from the VCP Table:

TOPT-EEB

1 • (1019) tests whether or not the identifier of a VCR
2 that can be retrieved from the VCP Table. A VCA
3 section is primarily selected within "Record_S".
4 Preferably, the selected VCA section is a section
5 which "AV_Checker_List" is equal to (or contains)
6 "RCV_AV_Checker_List". The identifier of a VCR can
7 be retrieved from the VCP Table, when said selected
8 VCA section comprises a VCR Identifier in its
9 "VCR_Id" field.

10 • If the identifier (called VCR_S_Id) of a VCR can be
11 retrieved from the VCP Table:

12 • Continues in step (1014) in order to retrieve,
13 from the VCR identified by VCR_S_Id, the
14 identifier of a VCA which has authority to build
15 a VC for the received file.

16 • If the identifier of a VCR cannot be retrieved from
17 the VCP Table:

18 • Continues in step (1016) in order to retrieve the
19 identifier (VCA_S_Id) of a Default VCA, from the
20 VCP Table (1020).

21 Advantages

22 • The present invention provides a faster and more efficient
23 way of detecting viruses, within a network device such as
24 an IP Router or a Firewall.

1 Computer program means or computer program in the present
2 context mean any expression, in any language, code or
3 notation, of a set of instructions intended to cause a system
4 having an information processing capability to perform a
5 particular function either directly or after conversion to
6 another language, code or notation and/or reproduction in a
7 different material form.

8 It is noted that the foregoing has outlined some of the more
9 pertinent objects and embodiments of the present invention.
10 This invention may be used for many applications. Thus,
11 although the description is made for particular arrangements
12 and methods, the intent and concept of the invention is
13 suitable and applicable to other arrangements and
14 applications. It will be clear to those skilled in the art
15 that other modifications to the disclosed embodiments can be
16 effected without departing from the spirit and scope of the
17 invention. The described embodiments ought to be construed to
18 be merely illustrative of some of the more prominent features
19 and applications of the invention. Other beneficial results
20 can be realized by applying the disclosed invention in a
21 different manner or modifying the invention in ways known to
22 those familiar with the art.